UNIVERSITY
of
GLASGOW

Gay, S.J. and Nagarajan, R. (2005) Communicating quantum processes. In, *Proceedings of the 32nd ACM SIGACT-SIGPLAN Symposium on Principles of Programming Languages, 12-14 January 2005*, pp. 145-157, Long Beach, California, USA.

http://eprints.gla.ac.uk/3475/

Glasgow ePrints Service
http://eprints.gla.ac.uk

Communicating Quantum Processes

Simon J. Gay
Department of Computing Science
University of Glasgow, UK
simon@dcs.gla.ac.uk

Rajagopal Nagarajan*
Department of Computer Science
University of Warwick, UK
biju@dcs.warwick.ac.uk

ABSTRACT

We define a language CQP (Communicating Quantum Processes) for modelling systems which combine quantum and classical communication and computation. CQP combines the communication primitives of the pi-calculus with primitives for measurement and transformation of quantum state; in particular, quantum bits (qubits) can be transmitted from process to process along communication channels. CQP has a static type system which classifies channels, distinguishes between quantum and classical data, and controls the use of quantum state. We formally define the syntax, operational semantics and type system of CQP, prove that the semantics preserves typing, and prove that typing guarantees that each qubit is owned by a unique process within a system. We illustrate CQP by defining models of several quantum communication systems, and outline our plans for using CQP as the foundation for formal analysis and verification of combined quantum and classical systems.

Categories and Subject Descriptors

D.3.1 [**Programming Languages**]: Formal Definitions and Theory; F.3.1 [**Logics and Meanings of Programs**]: Specifying and Verifying and Reasoning about Programs—*specification techniques*; F.3.1 [**Logics and Meanings of Programs**]: Semantics of Programming Languages—*operational semantics*

General Terms

Languages, Theory, Verification

Keywords

Formal language, quantum communication, quantum computing, semantics, types, verification

*Partially supported by the UK EPSRC (GR/S34090) and the EU Sixth Framework Programme (Project SecoQC).

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for pro£t or commercial advantage and that copies bear this notice and the full citation on the £rst page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior speci£c permission and/or a fee.
POPL'05, January 12–14, 2005, Long Beach, California, USA.
Copyright 2005 ACM 1-58113-830-X/05/0001 ...$5.00.

1. INTRODUCTION

Quantum computing and quantum communication have attracted growing interest since their inception as research areas more than twenty years ago, and there has been a surge of activity among computer scientists during the last few years. While quantum computing offers the prospect of vast improvements in algorithmic efficiency for certain problems, quantum cryptography can provide communication systems which will be secure even in the presence of hypothetical future quantum computers. As a practical technology, quantum communication has progressed far more rapidly than quantum computing. Secure communication involving quantum cryptography has recently been demonstrated in a scenario involving banking transactions in Vienna [21], systems are commercially available from Id Quantique, MagiQ Technologies and NEC, and plans have been reported to establish a nationwide quantum communication network in Singapore. Secure quantum communication will undoubtedly become a fundamental part of the technological infrastructure of society, long before quantum computers can tackle computations of a useful size.

However, secure quantum communication is not a solved problem. Although particular protocols have been mathematically proved correct (for example, Mayers' analysis [13] of the Bennett-Brassard protocol (BB84) [3] for quantum key distribution), this does not guarantee the security of systems which use them. Experience of classical security analysis has shown that even if *protocols* are theoretically secure, it is difficult to achieve robust and reliable implementations of secure *systems*: security can be compromised by flaws at the implementation level or at the boundaries between systems. To address this problem, computer scientists have developed an impressive armoury of techniques and tools for formal modelling, analysis and verification of classical security protocols and communication systems which use them [23]. These techniques have been remarkably successful both in establishing the security of new protocols and in demonstrating flaws in protocols which had previously been believed to be secure. Their strength lies in the ability to model *systems* as well as idealized protocols, and the flexibility to easily re-analyze variations in design.

Our research programme is to develop techniques and tools for formal modelling, analysis and verification of quantum communication and cryptographic systems. More precisely we aim to handle systems which combine quantum and classical communication and computation, for two reasons: the first quantum communication systems will implement communication between classical computers; and protocols

such as BB84 typically contain classical communication and computation as well as quantum cryptography. We cannot simply make use of existing techniques for classical security analysis: for example, treating the security of quantum cryptography axiomatically would not permit analysis of the protocols which *construct* quantum cryptographic keys. Furthermore, the inherently probabilistic nature of quantum systems means that not all verification consists of checking absolute properties; we need a probabilistic modelling and analysis framework.

Any formal analysis which involves automated tools requires a modelling language with a precisely-defined semantics. The purpose of this paper is to define a language, CQP (Communicating Quantum Processes), which will serve as the foundation for the programme described above. CQP combines the communication primitives of the pi-calculus [15, 25] with primitives for transformation and measurement of quantum state. In particular, qubits (quantum bits, the basic elements of quantum data) can be transmitted along communication channels. In Section 3 we introduce CQP through a series of examples which cover a wide spectrum of quantum information processing scenarios: a quantum coin-flipping game; a quantum communication protocol known as teleportation; and a quantum bit-commitment protocol. The latter will lead naturally to a model of the BB84 quantum key-distribution protocol in future work. In Section 4 we formalize the syntax of CQP and define an operational semantics which combines non-determinism (arising in the same way as in pi-calculus) with the probabilistic results of quantum measurements. In Section 5 we define a static type system which classifies data and communication channels, and crucially treats qubits as physical resources: if process P sends qubit q to process Q, then P must not access q subsequently, and this restriction can be enforced by static typechecking. In Section 6 we prove that the invariants of the type system are preserved by the operational semantics, guaranteeing in particular that at every point during execution of a system, every qubit is uniquely owned by a single parallel component. In Section 7 we outline our plans for further work, focusing on the use of both standard (non-deterministic) and probabilistic model-checking systems.

Related Work

There has been a great deal of interest in quantum programming languages, resulting in a number of proposals in different styles, for example [10, 18, 24, 26, 29]. Such languages can express arbitrary quantum state transformations and could be used to model quantum protocols in those terms. However, our view is that any model lacking an explicit treatment of communication is essentially incomplete for the analysis of protocols; certainly in the classical world, standard programming languages are not considered adequate frameworks in which to analyze or verify protocols. Nevertheless, Selinger's functional language QPL [26] in particular has influenced our choice of computational operators for CQP.

The closest work to our own, developed simultaneously but independently, is Jorrand and Lalire's QPAlg [9], which also combines communication in process calculus style with transformation and measurement of quantum state. The most distinctive features of our work are the type system and associated proofs, the explicit formulation of an expression language which can easily be extended, and our emphasis on a methodology for formal verification.

The work of Abramsky and Coecke [2] is also relevant. They define a category-theoretic semantic foundation for quantum protocols, which supports reasoning about systems and exposes deep connections between quantum systems and programming language semantics, but they do not define a formal syntax in which to specify models. It will be interesting to investigate the relationship between CQP and the semantic structures which they propose.

Acknowledgements We have benefitted from discussions with Philippe Jorrand, Marie Lalire and Nick Papanikolaou, and from the insightful comments of several referees.

2. PRELIMINARIES

We briefly introduce the aspects of quantum theory which are needed for the rest of the paper. For more detailed presentations we refer the reader to the books by Gruska [8] and Nielsen and Chuang [17]. Rieffel and Polak [22] give an account aimed at computer scientists.

A *quantum bit* or *qubit* is a physical system which has two basis states, conventionally written $|0\rangle$ and $|1\rangle$, corresponding to one-bit classical values. These could be, for example, spin states of a particle or polarization states of a photon, but we do not consider physical details. According to quantum theory, a general state of a quantum system is a *superposition* or linear combination of basis states. Concretely, a qubit has state $\alpha|0\rangle + \beta|1\rangle$, where α and β are complex numbers such that $|\alpha|^2 + |\beta|^2 = 1$; states which differ only by a (complex) scalar factor with modulus 1 are indistinguishable. States can be represented by column vectors:

$$\begin{pmatrix}\alpha\\ \beta\end{pmatrix} = \alpha|0\rangle + \beta|1\rangle.$$

Superpositions are illustrated by the quantum coin-flipping game which we discuss in Section 3.1. Formally, a quantum state is a unit vector in a Hilbert space, i.e. a complex vector space equipped with an inner product satisfying certain axioms. In this paper we will restrict attention to collections of qubits.

The basis $\{|0\rangle, |1\rangle\}$ is known as the *standard* basis. Other bases are sometimes of interest, especially the *diagonal* (or *dual*, or *Hadamard*) basis consisting of the vectors $|+\rangle = \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)$ and $|-\rangle = \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle)$. For example, with respect to the diagonal basis, $|0\rangle$ is in a superposition of basis states:

$$|0\rangle = \frac{1}{\sqrt{2}}|+\rangle + \frac{1}{\sqrt{2}}|-\rangle.$$

Evolution of a closed quantum system can be described by a *unitary transformation*. If the state of a qubit is represented by a column vector then a unitary transformation U can be represented by a complex-valued matrix (u_{ij}) such that $U^{-1} = U^*$, where U^* is the conjugate-transpose of U (i.e. element ij of U^* is $\bar{u}_{ji}$). U acts by matrix multiplication:

$$\begin{pmatrix}\alpha'\\ \beta'\end{pmatrix} = \begin{pmatrix}u_{00} & u_{01}\\ u_{10} & u_{11}\end{pmatrix}\begin{pmatrix}\alpha\\ \beta\end{pmatrix}$$

A unitary transformation can also be defined by its effect on basis states, which is extended linearly to the whole space.

For example, the *Hadamard* transformation is defined by

$$\begin{array}{rcl} |0\rangle & \mapsto & \frac{1}{\sqrt{2}}|0\rangle + \frac{1}{\sqrt{2}}|1\rangle \\ |1\rangle & \mapsto & \frac{1}{\sqrt{2}}|0\rangle - \frac{1}{\sqrt{2}}|1\rangle \end{array}$$

which corresponds to the matrix

$$\mathsf{H} = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}.$$

The Hadamard transformation creates superpositions:

$$\mathsf{H}|0\rangle = |+\rangle \qquad \mathsf{H}|1\rangle = |-\rangle.$$

We will also make use of the *Pauli* transformations, denoted by either $I, \sigma_x, \sigma_y, \sigma_z$ or $\sigma_0, \sigma_1, \sigma_2, \sigma_3$:

I or σ_0	σ_x or σ_1	σ_y or σ_2	σ_z or σ_3
$\begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$	$\begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}$	$\begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix}$	$\begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}$

A key feature of quantum physics is the role of *measurement*. If a qubit is in the state $\alpha|0\rangle + \beta|1\rangle$ then measuring its value gives the result 0 with probability $|\alpha|^2$ (leaving it in state $|0\rangle$) and the result 1 with probability $|\beta|^2$ (leaving it in state $|1\rangle$). Protocols sometimes specify measurement with respect to a different basis, such as the diagonal basis; this can be expressed as a unitary change of basis followed by a measurement with respect to the standard basis. Note that if a qubit is in state $|+\rangle$ then a measurement with respect to the standard basis gives result 0 (and state $|0\rangle$) with probability $\frac{1}{2}$, and result 1 (and state $|1\rangle$) with probability $\frac{1}{2}$. If a qubit is in state $|0\rangle$ then a measurement with respect to the diagonal basis gives result[1] 0 (and state $|+\rangle$) with probability $\frac{1}{2}$, and result 1 (and state $|-\rangle$)) with probability $\frac{1}{2}$, because of the representation of $|0\rangle$ in the diagonal basis noted above. If a classical bit is represented by a qubit using either the standard or diagonal basis, then a measurement with respect to the correct basis results in the original bit, but a measurement with respect to the other basis results in 0 or 1 with equal probability. This behaviour is used by the quantum bit-commitment protocol which we discuss in Section 3.3.

To go beyond single-qubit systems, we consider tensor products of spaces (in contrast to the cartesian products used in classical systems). If spaces U and V have bases $\{u_i\}$ and $\{v_j\}$ then $U \otimes V$ has basis $\{u_i \otimes v_j\}$. In particular, a system consisting of n qubits has a 2^n-dimensional space whose standard basis is $|00\ldots0\rangle \ldots |11\ldots1\rangle$. We can now consider measurements of single qubits or collective measurements of multiple qubits. For example, a 2-qubit system has basis $|00\rangle, |01\rangle, |10\rangle, |11\rangle$ and a general state is $\alpha|00\rangle + \beta|01\rangle + \gamma|10\rangle + \delta|11\rangle$ with $|\alpha|^2 + |\beta|^2 + |\gamma|^2 + |\delta|^2 = 1$. Measuring the first qubit gives result 0 with probability $|\alpha|^2 + |\beta|^2$ (leaving the system in state $\frac{1}{\sqrt{|\alpha|^2+|\beta|^2}}(\alpha|00\rangle + \beta|01\rangle)$) and result 1 with probability $|\gamma|^2 + |\delta|^2$ (leaving the system in state $\frac{1}{\sqrt{|\gamma|^2+|\delta|^2}}(\gamma|10\rangle + \delta|11\rangle)$); in each case we renormalize the state by multiplying by a suitable scalar factor. Measuring both qubits simultaneously gives result 0 with probability $|\alpha|^2$ (leaving the system in state $|00\rangle$), result 1 with probability $|\beta|^2$ (leaving the system in state $|01\rangle$) and so on; note that the association of basis states $|00\rangle, |01\rangle, |10\rangle, |11\rangle$ with results $0, 1, 2, 3$ is just a conventional choice. The power of quantum computing, in an algorithmic sense, results from calculating with superpositions of states; all the states in the superposition are transformed simultaneously (*quantum parallelism*) and the effect increases exponentially with the dimension of the state space. The challenge in quantum algorithm design is to make measurements which enable this parallelism to be exploited; in general this is very difficult.

We will make use of the *conditional not* (CNot) transformation on pairs of qubits. Its action on basis states is defined by

$$|00\rangle \mapsto |00\rangle \quad |01\rangle \mapsto |01\rangle \quad |10\rangle \mapsto |11\rangle \quad |11\rangle \mapsto |10\rangle$$

which can be understood as inverting the second qubit if and only if the first qubit is set, although in general we need to consider the effect on non-basis states.

Systems of two or more qubits can exhibit the phenomenon of *entanglement*, meaning that the states of the qubits are correlated. For example, consider a measurement of the first qubit of the state $\frac{1}{\sqrt{2}}(|00\rangle + |11\rangle)$. The result is 0 (and resulting state $|00\rangle$) with probability $\frac{1}{2}$, or 1 (and resulting state $|11\rangle$) with probability $\frac{1}{2}$. In either case a subsequent measurement of the second qubit gives a definite (non-probabilistic) result which is always the same as the result of the first measurement. This is true even if the entangled qubits are physically separated. Entanglement illustrates the key difference between the use of tensor product (in quantum systems) and cartesian product (in classical systems): an entangled state of two qubits is one which cannot be decomposed as a pair of single-qubit states. Entanglement is used in an essential way in the quantum teleportation protocol which we discuss in Section 3.2. That example uses the CNot transformation to create entanglement: $\mathsf{CNot}((\mathsf{H} \otimes I)|00\rangle) = \frac{1}{\sqrt{2}}(|00\rangle + |11\rangle)$.

3. EXAMPLES OF MODELLING IN CQP

3.1 A Quantum Coin-Flipping Game

Our first example is based on a scenario used by Meyer [14] to initiate the study of quantum game theory. Players P and Q play the following game: P places a coin, head upwards, in a box, and then the players take turns (Q, then P, then Q) to optionally turn the coin over, without being able to see it. Finally the box is opened and Q wins if the coin is head upwards.

Clearly neither player has a winning strategy, but the situation changes if the coin is a quantum system, represented by a qubit ($|0\rangle$ for head upwards, $|1\rangle$ for tail upwards). Turning the coin over corresponds to the transformation σ_1, and this is what P can do. But suppose that Q can apply H, which corresponds to transforming from head upwards ($|0\rangle$) to a superposition of head upwards and tail upwards ($\frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)$), and does this on both turns. Then we have two possible runs of the game, (a) and (b):

(a)

Action	State
	$\|0\rangle$
Q: H	$\frac{1}{\sqrt{2}}(\|0\rangle + \|1\rangle)$
P: σ_1	$\frac{1}{\sqrt{2}}(\|1\rangle + \|0\rangle)$
Q: H	$\|0\rangle$

(b)

Action	State
	$\|0\rangle$
Q: H	$\frac{1}{\sqrt{2}}(\|0\rangle + \|1\rangle)$
P: $-$	$\frac{1}{\sqrt{2}}(\|0\rangle + \|1\rangle)$
Q: H	$\|0\rangle$

[1]Strictly speaking, the outcome of the measurement is just the final state; the specific association of numerical results with final states is a matter of convention.

$$
\begin{array}{rcl}
P(s{:}\hat{}[\mathsf{Qbit},\hat{}[\mathsf{Qbit}]]) & = & s?[y{:}\mathsf{Qbit}, t{:}\hat{}[\mathsf{Qbit}]]\,.\,t![y]\,.\,\mathbf{0} \\
& | & s?[y{:}\mathsf{Qbit}, t{:}\hat{}[\mathsf{Qbit}]]\,.\,\{y *{=}\, \sigma_1\}\,.\,t![y]\,.\,\mathbf{0}
\end{array}
$$

$$
Q(x{:}\mathsf{Qbit}, s{:}\hat{}[\mathsf{Qbit},\hat{}[\mathsf{Qbit}]]) = \{x *{=}\, \mathsf{H}\}\,.\,(\mathsf{new}\ t{:}\hat{}[\mathsf{Qbit}])(s![x,t]\,.\,t?[z{:}\mathsf{Qbit}]\,.\,\{z *{=}\, \mathsf{H}\}\,.\,C(z))
$$

$$
\mathit{System}(x{:}\mathsf{Qbit}) = (\mathsf{new}\ s{:}\hat{}[\mathsf{Qbit},\hat{}[\mathsf{Qbit}]])(P(s) \mid Q(x,s))
$$

Figure 1: The quantum coin-flipping game in CQP

$$
\begin{array}{c}
x = |0\rangle\,;\emptyset\,;\mathit{System}(x) \\
\downarrow \quad \text{expand definition} \\
x = |0\rangle\,;\emptyset\,;(\mathsf{new}\ s{:}\hat{}[\mathsf{Qbit},\hat{}[\mathsf{Qbit}]])(P(s) \mid Q(x,s)) \\
\downarrow \quad \text{create channel } s \\
x = |0\rangle\,;s\,;P(s) \mid Q(x,s) \\
\downarrow \quad \text{expand definitions} \\
x = |0\rangle\,;s\,; \\
s?[y{:}\mathsf{Qbit}, t{:}\hat{}[\mathsf{Qbit}]]\,.\,t![y]\,.\,\mathbf{0} \mid s?[y{:}\mathsf{Qbit}, t{:}\hat{}[\mathsf{Qbit}]]\,.\,\{y *{=}\, \sigma_1\}\,.\,t![y]\,.\,\mathbf{0} \\
\mid \{x *{=}\, \mathsf{H}\}\,.\,(\mathsf{new}\ t{:}\hat{}[\mathsf{Qbit}])(s![x,t]\,.\,t?[z{:}\mathsf{Qbit}]\,.\,\{z *{=}\, \mathsf{H}\}\,.\,C(z) \\
\downarrow \quad \text{transform } x \\
x = \tfrac{1}{\sqrt{2}}(|0\rangle + |1\rangle)\,;s\,; \\
s?[y{:}\mathsf{Qbit}, t{:}\hat{}[\mathsf{Qbit}]]\,.\,t![y]\,.\,\mathbf{0} \mid s?[y{:}\mathsf{Qbit}, t{:}\hat{}[\mathsf{Qbit}]]\,.\,\{y *{=}\, \sigma_1\}\,.\,t![y]\,.\,\mathbf{0} \\
\mid (\mathsf{new}\ t{:}\hat{}[\mathsf{Qbit}])(s![x,t]\,.\,t?[z{:}\mathsf{Qbit}]\,.\,\{z *{=}\, \mathsf{H}\}\,.\,C(z)) \\
\downarrow \quad \text{create channel } t \\
x = \tfrac{1}{\sqrt{2}}(|0\rangle + |1\rangle)\,;s,t\,; \\
s?[y{:}\mathsf{Qbit}, t{:}\hat{}[\mathsf{Qbit}]]\,.\,t![y]\,.\,\mathbf{0} \mid s?[y{:}\mathsf{Qbit}, t{:}\hat{}[\mathsf{Qbit}]]\,.\,\{y *{=}\, \sigma_1\}\,.\,t![y]\,.\,\mathbf{0} \\
\mid s![x,t]\,.\,t?[z{:}\mathsf{Qbit}]\,.\,\{z *{=}\, \mathsf{H}\}\,.\,C(z) \\
\swarrow \qquad\qquad \searrow \text{communication} \\
\begin{array}{c}
x = \tfrac{1}{\sqrt{2}}(|0\rangle + |1\rangle)\,;s,t\,; \\
t![x]\,.\,\mathbf{0} \mid \mathit{Garbage} \\
\mid t?[z{:}\mathsf{Qbit}]\,.\,\{z *{=}\, \mathsf{H}\}\,.\,C(z) \\
\downarrow \\
x = \tfrac{1}{\sqrt{2}}(|0\rangle + |1\rangle)\,;s,t\,; \\
\mathbf{0} \mid \mathit{Garbage} \mid \{x *{=}\, \mathsf{H}\}\,.\,C(x) \\
\downarrow \\
x = |0\rangle\,;s,t\,;\mathit{Garbage} \mid C(x)
\end{array}
\qquad
\begin{array}{c}
x = \tfrac{1}{\sqrt{2}}(|0\rangle + |1\rangle)\,;s,t\,; \\
\mathit{Garbage} \mid \{x *{=}\, \sigma_1\}\,.\,t![x]\,.\,\mathbf{0} \\
\mid t?[z{:}\mathsf{Qbit}]\,.\,\{z *{=}\, \mathsf{H}\}\,.\,C(z) \\
\downarrow \text{transform } x \\
x = \tfrac{1}{\sqrt{2}}(|0\rangle + |1\rangle)\,;s,t\,; \\
\mathit{Garbage} \mid t![x]\,.\,\mathbf{0} \\
\mid t?[z{:}\mathsf{Qbit}]\,.\,\{z *{=}\, \mathsf{H}\}\,.\,C(z) \\
\downarrow \text{communication} \\
x = \tfrac{1}{\sqrt{2}}(|0\rangle + |1\rangle)\,;s,t\,; \\
\mathit{Garbage} \mid \mathbf{0} \mid \{x *{=}\, \mathsf{H}\}\,.\,C(x) \\
\downarrow \text{transform } x \\
x = |0\rangle\,;s,t\,;\mathit{Garbage} \mid C(x)
\end{array}
\end{array}
$$

Figure 2: Execution of the coin-flipping game

$$
\mathit{Alice}(x{:}\mathsf{Qbit}, c{:}\hat{}[0..3], z{:}\mathsf{Qbit}) = \{z, x *{=}\, \mathsf{CNot}\}\,.\,\{z *{=}\, \mathsf{H}\}\,.\,c![\mathsf{measure}\ z, x]\,.\,\mathbf{0}
$$

$$
\mathit{Bob}(y{:}\mathsf{Qbit}, c{:}\hat{}[0..3]) = c?[r{:}0..3]\,.\,\{y *{=}\, \sigma_r\}\,.\,\mathit{Use}(y)
$$

$$
\mathit{System}(x{:}\mathsf{Qbit}, y{:}\mathsf{Qbit}, z{:}\mathsf{Qbit}) = (\mathsf{new}\ c{:}\hat{}[0..3])(\mathit{Alice}(x,c,z) \mid \mathit{Bob}(y,c))
$$

Figure 3: Quantum teleportation in CQP

$$x,y,z = \tfrac{1}{\sqrt{2}}|001\rangle + \tfrac{1}{\sqrt{2}}|111\rangle\,;\emptyset\,;\mathit{System}(x,y,z)$$
$$\downarrow \quad \text{expand definition}$$
$$x,y,z = \tfrac{1}{\sqrt{2}}|001\rangle + \tfrac{1}{\sqrt{2}}|111\rangle\,;\emptyset\,;(\mathsf{new}\ c{:}\hat{}[\mathsf{0..3}])(\mathit{Alice}(x,c,z) \mid \mathit{Bob}(y,c))$$
$$\downarrow \quad \text{create channel } c$$
$$x,y,z = \tfrac{1}{\sqrt{2}}|001\rangle + \tfrac{1}{\sqrt{2}}|111\rangle\,;c\,;\mathit{Alice}(x,c,z) \mid \mathit{Bob}(y,c)$$
$$\downarrow \quad \text{expand definitions}$$
$$x,y,z = \tfrac{1}{\sqrt{2}}|001\rangle + \tfrac{1}{\sqrt{2}}|111\rangle\,;c\,;$$
$$\{z,x \mathrel{*}\!\!= \mathsf{CNot}\}\,.\,\{z \mathrel{*}\!\!= \mathsf{H}\}\,.\,c![\mathsf{measure}\ z,x]\,.\,\mathbf{0} \mid\ c?[r{:}\mathsf{0..3}]\,.\,\{y \mathrel{*}\!\!= \sigma_r\}\,.\,\mathit{Use}(y)$$
$$\downarrow \quad \text{permute } x,y,z;\ \text{transform } z,x$$
$$z,x,y = \tfrac{1}{\sqrt{2}}|110\rangle + \tfrac{1}{\sqrt{2}}|101\rangle\,;\,c\,;\{z \mathrel{*}\!\!= \mathsf{H}\}\,.\,c![\mathsf{measure}\ z,x]\,.\,\mathbf{0} \mid\ c?[r{:}\mathsf{0..3}]\,.\,\{y \mathrel{*}\!\!= \sigma_r\}\,.\,\mathit{Use}(y)$$
$$\downarrow \quad \text{transform } z$$
$$z,x,y = \tfrac{1}{2}|001\rangle + \tfrac{1}{2}|010\rangle - \tfrac{1}{2}|101\rangle - \tfrac{1}{2}|110\rangle\,;\,c\,;c![\mathsf{measure}\ z,x]\,.\,\mathbf{0} \mid c?[r{:}\mathsf{0..3}]\,.\,\{y \mathrel{*}\!\!= \sigma_r\}\,.\,\mathit{Use}(y)$$
$$\downarrow \quad \text{measure } z,x$$
$$\tfrac{1}{4} \bullet (z,x,y = |001\rangle; a,c;\ c![0]\,.\,\mathbf{0} \mid c?[r{:}\mathsf{0..3}]\,.\,\{y \mathrel{*}\!\!= \sigma_r\}\,.\,\mathit{Use}(y))$$
$$\boxplus \tfrac{1}{4} \bullet (z,x,y = |010\rangle; a,c;\ c![1]\,.\,\mathbf{0} \mid c?[r{:}\mathsf{0..3}]\,.\,\{y \mathrel{*}\!\!= \sigma_r\}\,.\,\mathit{Use}(y))$$
$$\boxplus \tfrac{1}{4} \bullet (z,x,y = |101\rangle; a,c;\ c![2]\,.\,\mathbf{0} \mid c?[r{:}\mathsf{0..3}]\,.\,\{y \mathrel{*}\!\!= \sigma_r\}\,.\,\mathit{Use}(y))$$
$$\boxplus \tfrac{1}{4} \bullet (z,x,y = |110\rangle; a,c;\ c![3]\,.\,\mathbf{0} \mid c?[r{:}\mathsf{0..3}]\,.\,\{y \mathrel{*}\!\!= \sigma_r\}\,.\,\mathit{Use}(y))$$

$\frac{1}{4}\downarrow$	$\frac{1}{4}\downarrow$	$\frac{1}{4}\downarrow$	$\frac{1}{4}\downarrow$
$z,x,y = \lvert001\rangle\,;c\,;$	$z,x,y = \lvert010\rangle\,;c\,;$	$z,x,y = \lvert101\rangle\,;c\,;$	$z,x,y = \lvert110\rangle\,;c\,;$
$c![0]\,.\,\mathbf{0} \mid c?[r{:}\mathsf{0..3}]\,.$	$c![1]\,.\,\mathbf{0} \mid c?[r{:}\mathsf{0..3}]\,.$	$c![2]\,.\,\mathbf{0} \mid c?[r{:}\mathsf{0..3}]\,.$	$c![3]\,.\,\mathbf{0} \mid c?[r{:}\mathsf{0..3}]\,.$
$\{y \mathrel{*}\!\!= \sigma_r\}\,.\,\mathit{Use}(y)$	$\{y \mathrel{*}\!\!= \sigma_r\}\,.\,\mathit{Use}(y)$	$\{y \mathrel{*}\!\!= \sigma_r\}\,.\,\mathit{Use}(y)$	$\{y \mathrel{*}\!\!= \sigma_r\}\,.\,\mathit{Use}(y)$
$\downarrow$	$\downarrow$	$\downarrow$	$\downarrow$ communication
$z,x,y = \lvert001\rangle\,;c\,;$	$z,x,y = \lvert010\rangle\,;c\,;$	$z,x,y = \lvert101\rangle\,;c\,;$	$z,x,y = \lvert110\rangle\,;c\,;$
$\{y \mathrel{*}\!\!= \sigma_0\}\,.\,\mathit{Use}(y)$	$\{y \mathrel{*}\!\!= \sigma_1\}\,.\,\mathit{Use}(y)$	$\{y \mathrel{*}\!\!= \sigma_2\}\,.\,\mathit{Use}(y)$	$\{y \mathrel{*}\!\!= \sigma_3\}\,.\,\mathit{Use}(y)$
$\downarrow$	$\downarrow$	$\downarrow$	$\downarrow$ transform y
$z,x,y = \lvert001\rangle\,;c\,;$	$z,x,y = \lvert011\rangle\,;c\,;$	$z,x,y = -i\lvert101\rangle\,;c\,;$	$z,x,y = -\lvert111\rangle\,;c\,;$
$\mathit{Use}(y)$	$\mathit{Use}(y)$	$\mathit{Use}(y)$	$\mathit{Use}(y)$

Figure 4: Execution of the quantum teleportation protocol

and in each case the coin finishes head upwards. To verify this we calculate that the state $\frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)$ is invariant under σ_1:

$$\begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \frac{1}{\sqrt{2}} \begin{pmatrix} 1 \\ 1 \end{pmatrix} = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 \\ 1 \end{pmatrix}$$

and that the Hadamard transformation H is self-inverse:

$$\frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix} \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix} = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$$

Meyer considers game-theoretic issues relating to the expected outcome of repeated runs, but we just model a single run in CQP (Figure 1). Most of the syntax of CQP is based on typed pi-calculus, using fairly common notation (for example, see Pierce and Sangiorgi's presentation [20]). P and Q communicate by means of the typed channel $s{:}\hat{}[\mathsf{Qbit},\hat{}[\mathsf{Qbit}]]$ which carries qubits. It is a parameter of both P and Q. At the top level, *System* creates s with $(\mathsf{new}\ s{:}\hat{}[\mathsf{Qbit},\hat{}[\mathsf{Qbit}]])$ and starts P and Q in parallel. Q and *System* are also parameterized by x, the qubit representing the coin in its initial state.

Q applies $(x \mathrel{*}\!\!= \mathsf{H})$ the Hadamard transformation to x; this syntax is based on Selinger's QPL [26]. This expression is converted into an action by $\{\ldots\}$. Using a standard pi-calculus programming style, Q creates a channel t and sends $(s![x,t])$ it to P along with the qubit x. P will use t to send the qubit back, and Q receives it with $t?[z{:}\mathsf{Qbit}]$, binding it to the name z in the rest of the code. Finally Q applies H again, and continues with some behaviour $C(z)$.

P contains two branches of behaviour, corresponding to the possibilities of applying (second branch) or not applying (first branch) the transformation σ_1. Both branches terminate with the null process $\mathbf{0}$. The branches are placed in parallel[2] and the operational semantics means that only one of them interacts with Q. The other branch takes no further part in the execution of the system, because there is nothing for it to interact with; in Figure 2 (see below) we call it *Garbage*.

Figure 2 shows the execution (combining some steps) of *System* according to the operational semantics which we will define formally in Section 4. Reduction takes place on configurations $(\sigma;\phi;P)$ where σ is a list of qubits and their collective state, ϕ lists the channels which have been created, and P is a process term. Note that the state of the qubits *must* be a global property in order to be physically realistic. We record the channels globally in order to give the semantics a uniform style; this is different from the usual approach to pi-calculus semantics, but (modulo garbage collection) is equivalent to expanding the scope of every new before beginning execution.

The execution of *System* tracks the informal calculation which we worked through above. Our CQP model makes the manipulation of the qubit very explicit; there are other ways to express the behaviour (including putting everything into a single process with no communication), but the point is that we have a framework in which to discuss such issues.

[2]Simpler definitions can be obtained if we add guarded sums to CQP; there is then no need for the channel t. This is straightforward but we have chosen instead to simplify the presentation of the semantics.

$$Alice'(s{:}\hat{}[\mathsf{Qbit}], c{:}\hat{}[0..3], z{:}\mathsf{Qbit}) = s?[x{:}\mathsf{Qbit}] \,.\, Alice(x, c, z)$$

$$Bob'(t{:}\hat{}[\mathsf{Qbit}], c{:}\hat{}[0..3]) = t?[y{:}\mathsf{Qbit}] \,.\, Bob(y, c)$$

$$Source(s{:}\hat{}[\mathsf{Qbit}], t{:}\hat{}[\mathsf{Qbit}]) = (\mathsf{qbit}\ x, y)(\{x *{=}\ \mathsf{H}\} \,.\, \{x, y *{=}\ \mathsf{CNot}\} \,.\, s![x] \,.\, t![y] \,.\, \mathbf{0})$$

$$System'(z{:}\mathsf{Qbit}) = (\mathsf{new}\ c{:}\hat{}[0..3], s{:}\hat{}[\mathsf{Qbit}], t{:}\hat{}[\mathsf{Qbit}])(Alice'(s, c, z) \mid Bob'(t, c) \mid Source(s, t))$$

Figure 5: Quantum teleportation with an EPR source

3.2 Quantum Teleportation

The quantum teleportation protocol [4] is a procedure for transmitting a quantum state via a non-quantum medium. This protocol is particularly important: not only is it a fundamental component of several more complex protocols, but it is likely to be a key enabling technology for the development of the *quantum repeaters* [6] which will be necessary in large-scale quantum communication networks.

Figure 3 shows a simple model of the quantum teleportation protocol. Alice and Bob each possess one qubit (x for Alice, y for Bob) of an entangled pair whose state is $\frac{1}{\sqrt{2}}(|00\rangle + |11\rangle)$. At this point we are assuming that appropriate qubits will be supplied to Alice and Bob as parameters of the system. Alice is also parameterized by a qubit z, whose state is to be teleported. She applies ($z, x{*}{=}\mathsf{CNot}$) the conditional not transformation to z and x and then applies ($z *{=}\ \mathsf{H}$) the Hadamard transformation to z, finally measuring z and x to yield a two-bit classical value which she sends ($c![\mathsf{measure}\ z, x]$) to Bob on the typed channel $c{:}\hat{}[0..3]$ and then terminates ($\mathbf{0}$). Bob receives ($c?[r{:}0..3]$) this value and uses it to select[3] a *Pauli* transformation $\sigma_0 \ldots \sigma_3$ to apply ($y *{=} \sigma_r$) to y. The result is that Bob's qubit y takes on the state of z, without a physical qubit having been transmitted from Alice to Bob. Bob may then use y in his continuation process $Use(y)$.

This example introduces measurement, with a syntax similar to that of Selinger's QPL [26]. We treat measurement as an expression, executed for its value as well as its side-effect on the quantum state. Because the result of a measurement is probabilistic, evaluation of a measure expression introduces a probability distribution over configurations: $\boxplus_{0 \leqslant i \leqslant n} p_i \bullet (\sigma_i; \phi_i; P_i)$. The next step is a probabilistic transition to one of the configurations; no reduction takes place underneath a probability distribution. In general a configuration reduces non-deterministically to one of a collection of probability distributions over configurations (in some cases this is trivial, with only one distribution or only one configuration within a distribution). A non-trivial probability distribution makes a probabilistic transition to a single configuration; this step is omitted in the case of a trivial distribution.

Figure 4 shows the complete execution of *System* in the particular case in which z, the qubit being teleported, has state $|1\rangle$. The measurement produces a probability distribution over four configurations, but in all cases the final configuration (process $Use(y)$) has a state consisting of a single basis vector in which $y = |1\rangle$. To verify the protocol for an arbitrary qubit, we can repeat the calculation with initial state $x, y, z = \frac{1}{\sqrt{2}}(|00\rangle + |11\rangle) \otimes (\alpha|0\rangle + \beta|1\rangle)$.

Alice and Bob are parameterized by their parts (x, y) of the entangled pair (and by the channel c). We can be more explicit about the origin of the entangled pair by introducing what is known in the physics literature as an *EPR source*[4] (computer scientists might regard it as an *entanglement server*). This process constructs the entangled pair (by using the Hadamard and controlled not transformations; note that our semantics (Section 4) specifies that the qubits x and y are each initialized to $|0\rangle$) and sends its components to Alice and Bob on the typed channels $s, t{:}\hat{}[\mathsf{Qbit}]$. Figure 5 shows the revised model.

3.3 Bit-Commitment

The bit-commitment problem is to design a protocol such that Alice chooses a one-bit value which Bob then attempts to guess. The key issue is that Alice must evaluate Bob's guess with respect to her original choice of bit, without changing her mind; she must be committed to her choice. Similarly, Bob must not find out Alice's choice before making his guess. Bit-commitment turns out to be an important primitive in cryptographic protocols. Classical bit-commitment schemes rely on assumptions on the computational complexity of certain functions; it is natural to ask whether quantum techniques can remove these assumptions.

We will discuss a quantum bit-commitment protocol due to Bennett and Brassard [3] which is closely related to the quantum key-distribution protocol proposed in the same paper and known as BB84. The following description of the protocol is based on Gruska's [8] presentation.

1. Alice randomly chooses a bit x and a sequence of bits xs. She encodes xs as a sequence of qubits and sends them to Bob. This encoding uses the standard basis (representing 0 by $|0\rangle$ and 1 by $|1\rangle$) if $x = 0$, and the diagonal basis (representing 0 by $|+\rangle$ and 1 by $|-\rangle$) if $x = 1$.

2. Upon receiving each qubit, Bob randomly chooses to measure it with respect to either the standard basis or the diagonal basis. For each measurement he stores the result and his choice of basis. If the basis he chose matches Alice's x then the result of the measurement is the same as the corresponding bit from xs; if not, then the result is 0 or 1 with equal probability. After receiving all of the qubits, Bob tells Alice his guess at the value of x.

3. Alice tells Bob whether or not he guessed correctly. To certify her claim she sends xs to Bob.

[3] We can easily extend the expression language of CQP to allow explicit testing of r.

[4] EPR stands for Einstein, Podolsky and Rosen.

$$
\begin{aligned}
&\mathit{Alice}(x:\mathsf{Bit}, xs:\mathsf{Bit\ List}, c:\hat{}[\mathsf{Qbit}], d:\hat{}[\mathsf{Bit}], e:\hat{}[\mathsf{Int}], f:\hat{}[\mathsf{Bit\ List}]) = \\
&\quad e![\mathsf{length}(xs)]\,.\,\mathit{AliceSend}(x, \mathsf{length}(xs), xs, xs, c, d, e, f)
\end{aligned}
$$

$$
\begin{aligned}
&\mathit{AliceSend}(x:\mathsf{Bit}, n:\mathsf{Int}, xs:\mathsf{Bit\ List}, ys:\mathsf{Bit\ List}, c:\hat{}[\mathsf{Qbit}], d:\hat{}[\mathsf{Bit}], e:\hat{}[\mathsf{Int}], f:\hat{}[\mathsf{Bit\ List}]) = \\
&\quad \mathsf{if}\ n = 0\ \mathsf{then}\ \mathit{AliceReceive}(x, \mathsf{length}(ys), ys, c, d, e, f) \\
&\quad \mathsf{else}\ (\mathsf{qbit}\ q)(\ \{\mathsf{if}\ \mathsf{hd}(xs) = 1\ \mathsf{then}\ q *= \sigma_x\ \mathsf{else\ unit}\}\,.\,\{\mathsf{if}\ x = 1\ \mathsf{then}\ q *= \mathsf{H}\ \mathsf{else\ unit}\}\,.\,c![q]\,. \\
&\qquad\qquad \mathit{AliceSend}(x, n-1, \mathsf{tl}(xs), ys, c, d, e, f))
\end{aligned}
$$

$$
\mathit{AliceReceive}(x:\mathsf{Bit}, n:\mathsf{Int}, ys:\mathsf{Bit\ List}, d:\hat{}[\mathsf{Bit}], f:\hat{}[\mathsf{Bit\ List}]) = d?[g:\mathsf{Bit}]\,.\,d![x]\,.\,f![ys]\,.\,\mathbf{0}
$$

$$
\mathit{Bob}(c:\hat{}[\mathsf{Qbit}], d:\hat{}[\mathsf{Bit}], e:\hat{}[\mathsf{Int}], f:\hat{}[\mathsf{Bit\ List}], r:\hat{}[\mathsf{Bit}]) = e?[n:\mathsf{Int}]\,.\,\mathit{BobReceive}([\,], n, c, d, f, r)
$$

$$
\begin{aligned}
&\mathit{BobReceive}(m:(\mathsf{Bit} * \mathsf{Bit})\ \mathsf{List}, n:\mathsf{Int}, c:\hat{}[\mathsf{Qbit}], d:\hat{}[\mathsf{Bit}], e:\hat{}[\mathsf{Int}], f:\hat{}[\mathsf{Bit\ List}], r:\hat{}[\mathsf{Bit}]) = \\
&\quad \mathsf{if}\ n = 0\ \mathsf{then}\ r?[g:\mathsf{Bit}]\,.\,d![g]\,.\,d?[a:\mathsf{Bit}]\,.\,f?[vs:\mathsf{Bit\ List}]\,.\,\mathit{BobVerify}(m, vs, a, \mathsf{length}(m)) \\
&\quad \mathsf{else}\ c?[x:\mathsf{Qbit}]\,.\,r?[y:\mathsf{Bit}]\,.\,\{\mathsf{if}\ y = 1\ \mathsf{then}\ x *= \mathsf{H}\ \mathsf{else\ unit}\}\,.\,\mathit{BobReceive}(m@[(y, \mathsf{measure}\ x)], n-1, c, d, r)
\end{aligned}
$$

$$
\begin{aligned}
&\mathit{BobVerify}(m:(\mathsf{Bit} * \mathsf{Bit})\ \mathsf{List}, vs:\mathsf{Bit\ List}, a:\mathsf{Bit}, n:\mathsf{Int}) = \\
&\quad \mathsf{if}\ n = 0\ \mathsf{then}\ \mathit{Verified} \\
&\quad \mathsf{else\ if}\ \mathsf{fst}(\mathsf{hd}(m)) = a\ \mathsf{then} \\
&\qquad\qquad \mathsf{if}\ \mathsf{snd}(\mathsf{hd}(m)) = \mathsf{hd}(vs)\ \mathsf{then}\ \mathit{BobVerify}(\mathsf{tl}(m), \mathsf{tl}(vs), a, n-1) \\
&\qquad\qquad \mathsf{else}\ \mathit{NotVerified} \\
&\quad \mathsf{else}\ \mathit{BobVerify}(\mathsf{tl}(m), \mathsf{tl}(vs), a, n-1)
\end{aligned}
$$

$$
\mathit{Random}(r:\hat{}[\mathsf{Bit}]) = (\mathsf{qbit}\ q)(\{q *= \mathsf{H}\}\,.\,r![\mathsf{measure}\ q]\,.\,\mathit{Random}(r))
$$

$$
\begin{aligned}
&\mathit{System}(x:\mathsf{Bit}, xs:\mathsf{Bit\ List}) = \\
&\quad (\mathsf{new}\ c:\hat{}[\mathsf{Qbit}], d:\hat{}[\mathsf{Bit}], e:\hat{}[\mathsf{Int}], f:\hat{}[\mathsf{Bit\ List}], r:\hat{}[\mathsf{Bit}])(\mathit{Alice}(x, xs, c, d, e, f) \mid \mathit{Bob}(c, d, e, f, r) \mid \mathit{Random}(r))
\end{aligned}
$$

Figure 6: Quantum bit-commitment in CQP

4. Bob verifies Alice's claim by looking at the measurements in which he used the basis corresponding to x, and checking that the results are the same as the corresponding bits from xs. He can also check that the results of the other measurements are sufficiently random (i.e. not significantly correlated with the corresponding bits from xs).

Figure 6 shows our model of this protocol in CQP. The complexity of the definitions reflects the fact that we have elaborated much of the computation which is implicit in the original description. The definitions use the following features which are not present in our formalization of CQP, but can easily be added.

- The type constructor List and associated functions and constructors such as hd, tl, length, [ ], @.
- Product types ($*$) and functions such as fst, snd.
- if $-$ then $-$ else for expressions and processes.
- Recursive process definitions.

Alice is parameterized by x and xs; they could be explicitly chosen at random if desired. In *AliceSend*, the encoding of xs relies on the fact that (qbit q) initializes q to $|0\rangle$. *Bob* uses m to record the results of his measurements, and n (received from *Alice* initially) as a recursion parameter. *Bob* receives random bits, for his choices of basis, from the server *Random*; he also guesses x randomly. The state *BobVerify* carries out the first part of step (4) above, but we have not included a check for non-correlation of the remaining bits. The states *Verified* and *NotVerified* stand for whatever action Bob takes after discovering whether or not Alice's statement in step (3) is true.

All measurement in CQP is with respect to the standard basis. We express measurements with respect to other bases by first applying a unitary transformation corresponding to a change of basis. This can be seen in the else branch of *BobReceive*, where the code $\{\mathsf{if}\ y = 1\ \mathsf{then}\ x *= \mathsf{H}\ \mathsf{else\ unit}\}$ applies a change of basis if necessary.

Communication between *Alice* and *Bob* uses four separate channels, $c, \ldots, f$. This proliferation of channels is a consequence of the fact that our type system associates a unique message type with each channel. Introducing *session types* [27] would allow a single channel to be used for the entire protocol, although it is worth noting that depending on the physical implementation of qubits, separation of classical and quantum channels might be the most accurate model.

We intend to use this CQP model as the basis for various kinds of formal analysis of the bit-commitment protocol; we make some specific suggestions in Section 7. We should point out, however, that this bit-commitment protocol is insecure in that it allows Alice to cheat: if each qubit which she sends to Bob is part of an entangled pair, then Bob's measurements transmit information back to Alice which she can use to change x after receiving Bob's guess. The real value of quantum bit-commitment is as a stepping-stone to the BB84 quantum key-distribution protocol, which has a very similar structure and is already being used in practical quantum communication systems.

$$
\begin{array}{rcl}
T & ::= & \mathsf{Int} \mid \mathsf{Unit} \mid \mathsf{Qbit} \mid \hat{}[\widetilde{T}] \mid \mathsf{Op}(1) \mid \mathsf{Op}(2) \mid \ldots \\
v & ::= & x \mid \mathsf{0} \mid \mathsf{1} \mid \ldots \mid \mathsf{unit} \mid \mathsf{H} \mid \ldots \\
e & ::= & v \mid \mathsf{measure}\ \widetilde{e} \mid \widetilde{e} \mathrel{*}= e \mid e{+}e \\
P & ::= & \mathbf{0} \mid (P \mid P) \mid e?[\widetilde{x}{:}\widetilde{T}]\,.\,P \mid e![\widetilde{e}]\,.\,P \mid \{e\}\,.\,P \mid (\mathsf{new}\ x{:}T)P \mid (\mathsf{qbit}\ x)P
\end{array}
$$

Figure 7: Syntax of CQP

$$
\begin{array}{rcl}
v & ::= & \ldots \mid q \mid c \\
E & ::= & [\,] \mid \mathsf{measure}\ E, \widetilde{e} \mid \mathsf{measure}\ v, E, \widetilde{e} \mid \ldots \mid \mathsf{measure}\ \widetilde{v}, E \mid E, \widetilde{e} \mathrel{*}= e \mid v, E, \widetilde{e} \mathrel{*}= e \\
 & & \mid \ldots \mid \widetilde{v} \mathrel{*}= E \mid E{+}e \mid v{+}E \\
F & ::= & [\,]?[\widetilde{x}{:}\widetilde{T}]\,.\,P \mid [\,]![\widetilde{e}]\,.\,P \mid v![[\,], \widetilde{e}]\,.\,P \mid v![v, [\,], \widetilde{e}]\,.\,P \mid \ldots \mid v![\widetilde{v}, [\,]]\,.\,P \mid \{[\,]\}\,.\,P
\end{array}
$$

Figure 8: Internal syntax of CQP

4. SYNTAX AND SEMANTICS

We now formally define the syntax and operational semantics of the core of CQP, excluding named process definitions and recursion, which can easily be added.

4.1 Syntax

The syntax of CQP is defined by the grammar in Figure 7. Types T consist of data types such as Int and Unit (others can easily be added), the type Qbit of qubits, channel types $\hat{}[T_1, \ldots, T_n]$ (specifying that each message is an n-tuple with component types $T_1, \ldots, T_n$) and operator types $\mathsf{Op}(n)$ (the type of a unitary operator on n qubits). The integer range type 0..3 used in the teleportation example is purely for clarification and should be replaced by Int; we do not expect to typecheck with range types.

We use the notation $\widetilde{T} = T_1, \ldots, T_n$ and $\widetilde{e} = e_1, \ldots, e_n$ and write $|\widetilde{e}|$ for the length of a tuple. Values v consist of variables (x, y, z etc.), literal values of data types ($0, 1, \ldots$ and unit) and unitary operators such as the Hadamard operator H. Expressions e consist of values, measurements $\mathsf{measure}\ e_1, \ldots, e_n$, applications $e_1, \ldots, e_n \mathrel{*}= e$ of unitary operators, and expressions involving data operators such as $e + e'$ (others can easily be added). Note that although the syntax refers to measurements and transformation of expressions e, the type system will require these expressions to refer to qubits. Processes P consist of the null (terminated) process $\mathbf{0}$, parallel compositions $P|Q$, inputs $e?[\widetilde{x}{:}\widetilde{T}]\,.\,P$ (notation: $\widetilde{x}{:}\widetilde{T} = x_1{:}T_1, \ldots, x_n{:}T_n$, declaring the types of all the input-bound variables), outputs $e![\widetilde{e}]\,.\,P$, actions $\{e\}\,.\,P$ (typically e will be an application of a unitary operator), channel declarations $(\mathsf{new}\ x : T)P$ and qubit declarations $(\mathsf{qbit}\ x)P$. In inputs and outputs, the expression e will be constrained by the type system to refer to a channel.

The grammar in Figure 8 defines the *internal* syntax of CQP, which is needed in order to define the operational semantics. Values are extended by two new forms: qubit names q, and channel names c. Evaluation contexts $E[\,]$ (for expressions) and $F[\,]$ (for processes) are used in the definition of the operational semantics, in the style of Wright and Felleisen [30]. The structure of $E[\,]$ is used to define call-by-value evaluation of expressions; the hole $[\,]$ specifies the first part of the expression to be evaluated. The structure of $F[\,]$ is used to define reductions of processes, specifying which expressions within a process must be evaluated.

Given a process P we define its free variables $fv(P)$, free qubit names $fq(P)$ and free channel names $fc(P)$ as usual; the binders (of x or $\widetilde{x}$) are $y?[\widetilde{x}{:}\widetilde{T}]$, $(\mathsf{qbit}\ x)$ and $(\mathsf{new}\ x{:}T)$.

4.2 Operational Semantics

The operational semantics of CQP is defined by reductions (small-step evaluations of expressions, or inter-process communications) and probabilistic transitions. The general form of a reduction is $t \longrightarrow \boxplus_i\ p_i \bullet t_i$ where t and the t_i are configurations consisting of expressions or processes with state information. The notation $\boxplus_i\ p_i \bullet t_i$ denotes a probability distribution over configurations, in which $\Sigma_i p_i = 1$; we may also write this distribution as $p_1 \bullet t_1 \boxplus \cdots \boxplus p_n \bullet t_n$. If the probability distribution contains a single configuration (with probability 1) then we simply write $t \longrightarrow t'$. Probability distributions reduce probabilistically to single configurations: $\boxplus_i\ p_i \bullet t_i \xrightarrow{p_i} t_i$ (with probability p_i, the distribution $\boxplus_i\ p_i \bullet t_i$ reduces to t_i).

This separation of reductions and probabilistic transitions is necessary because of non-determinisim. A general state of a process may have a number of possible reductions, arising from communication or evaluation of expressions, as well as a possible measurement step with a probabilistic outcome. In order to meaningfully interpret the probabilities in such a state, we must decide whether to resolve the non-determinism or the probability first. We have chosen to resolve the non-determinism first; thus, one of the non-deterministic reductions is the measurement, resulting in a probability distribution over configurations. This means that our semantics is consistent with the PRISM probabilistic model-checker [12], which we intend to use for verification. Cazorla *et al.* [5] discuss this issue further, and survey the approaches taken by several authors.

The semantics of expressions is defined by the reduction relations $\longrightarrow_{\mathsf{v}}$ and $\longrightarrow_{\mathsf{e}}$ (Figure 10), both on configurations of the form $(\sigma; \phi; e)$. If n qubits have been declared then σ has the form $q_0, \ldots, q_{n-1} = |\psi\rangle$ where $|\psi\rangle = \alpha_0|\psi_0\rangle + \cdots + \alpha_{2^n-1}|\psi_{2^n-1}\rangle$ is an element of the 2^n-dimensional vector space with basis $|\psi_0\rangle = |0\ldots0\rangle, \ldots, |\psi_{2^n-1}\rangle = |1\ldots1\rangle$. The remaining part of the configuration, ϕ, is a list of channel names, which plays little part in the semantics but allows bookkeeping lemmas to be proved. Reductions $\longrightarrow_{\mathsf{v}}$ are basic steps of evaluation, defined by the rules R-Plus (and similar rules for any other data operators), R-Measure and R-Trans. Rule R-Perm allows qubits in the state to be permuted, compensating for the way that R-Measure and R-Trans operate on qubits listed first in the state. Mea-

(S-Nil) $P \mid \mathbf{0} \equiv P$ (S-Comm) $P \mid Q \equiv Q \mid P$ (S-Assoc) $P \mid (Q \mid R) \equiv (P \mid Q) \mid R$

Figure 9: Structural congruence

surement specifically measures the values of a collection of qubits; in the future we should generalize to measuring *observables* as allowed by quantum physics.

Reductions $\longrightarrow_{\mathsf{e}}$ extend execution to evaluation contexts $E[\,]$, as defined by rule R-Context. Note that the probability distribution remains at the top level.

Figure 11 defines the reduction relation $\longrightarrow$ on configurations of the form $(\sigma; \phi; P)$. Rule R-Expr lifts reductions of expressions to $F[\,]$ contexts, again keeping probability distributions at the top level. Rule R-Com defines communication in the style of pi-calculus, making use of substitution, which is defined in the usual way (we assume that bound identifiers are renamed to avoid capture). Rule R-Act trivially removes actions; in general the reduction of the action expression to v will have involved side-effects such as measurement or transformation of quantum state. Rules R-New and R-Qbit create new channels and qubits, updating the state information in the configuration; qubits are initialized to $|0\rangle$. Note that this treatment of channel creation is different from standard presentations of the pi-calculus; we treat both qubits and channels as elements of a global store. Rule R-Par allows reduction to take place in parallel contexts, again lifting the probability distribution to the top level, and rule R-Cong allows the use of a structural congruence relation as in the pi-calculus. Structural congruence is the smallest congruence relation (closed under the process constructions) containing α-equivalence (with respect to the binders defined in Section 4.1) and closed under the rules in Figure 9.

5. TYPE SYSTEM

The typing rules defined in Figure 12 apply to the syntax defined in Figure 7. Environments Γ are mappings from variables to types in the usual way. Typing judgements are of two kinds. $\Gamma \vdash e : T$ means that expression e has type T in environment Γ. $\Gamma \vdash P$ means that process P is well-typed in environment Γ. The rules for expressions are straightforward; note that in rule T-Trans, $x_1, \ldots, x_n$ must be distinct variables of type Qbit.

In rule T-Par the operation $+$ on environments (Definition 1) is the key to ensuring that each qubit is controlled by a unique part of a system. The hypothesis that $\Gamma_1 + \Gamma_2$ must be defined means that it is not possible to type a system in which a qubit is shared by parallel components. This is very similar to the linear type system for the pi-calculus, defined by Kobayashi *et al.* [11].

Definition 1 (Addition of Environments). *The partial operation of adding a typed variable to an environment, $\Gamma + x{:}T$, is defined by*

$$\begin{array}{lcll} \Gamma + x{:}T & = & \Gamma, x{:}T & \text{if } x \notin dom(\Gamma) \\ \Gamma + x{:}T & = & \Gamma & \text{if } T \neq \mathsf{Qbit} \text{ and } x{:}T \in \Gamma \\ \Gamma + x{:}T & = & \text{undefined, otherwise} & \end{array}$$

This operation is extended inductively to a partial operation $\Gamma + \Delta$ on environments.

Rule T-Out allows output of classical values and qubits to be combined, but the qubits must be distinct variables and they cannot be used by the continuation of the outputting process (note the hypothesis $\Gamma \vdash P$). The remaining rules are straightforward.

According to the operational semantics, execution of qbit and new declarations introduces qubit names and channel names. In order to be able to use the type system to prove results about the behaviour of executing processes, we introduce the internal type system (Figure 13). This uses judgements $\Gamma; \Sigma; \Phi \vdash e : T$ and $\Gamma; \Sigma; \Phi \vdash P$ where Σ is a set of qubit names and Φ is a mapping from channel names to channel types. Most of the typing rules are straightforward extensions of the corresponding rules in Figure 12. Because references to qubits may now be either variables or explicit qubit names, the rules represent them by general expressions e and impose conditions that e is either a variable or a qubit name. This is seen in rules IT-Trans and IT-Out. Rule IT-Par is similar to T-Par in enforcing non-sharing of qubits, and is generalized to cover qubit names as well as variables.

By standard techniques for linear type systems, the typing rules in Figure 12 can be converted into a typechecking algorithm for CQP models.

As an illustration of the linear control of qubits, consider the coin-flipping example (Figure 1). In P, any non-trivial continuation replacing $\mathbf{0}$ would not be able to use the qubit y, which has been sent on t. In Q, after the qubit x has been sent on s, the continuation cannot use x. Of course, at run-time, the qubit variable z in $t?[z{:}\mathsf{Qbit}]$ is instantiated by x, but that is not a problem because P does not use x after sending it. In *System*, x is used as an actual parameter of Q and therefore could not also be used as an actual parameter of P (if P had a formal parameter of type Qbit).

6. SOUNDNESS OF THE TYPE SYSTEM

We prove a series of standard lemmas, following the approach of Wright and Felleisen [30], leading to a proof that typing is preserved by execution of processes (Theorem 1). We then prove that in a typable process, each qubit is used by at most one of any parallel collection of sub-processes (Theorem 2); because of type preservation, this property holds at every step of the execution of a typable process. This reflects the physical reality of the protocols which we want to model. It is similar to the unique ownership theorem of Ennals *et al.* [7].

We can also prove a standard runtime safety theorem, stating that a typable process generates no communication errors or incorrectly-applied operators, but we have not included it in the present paper.

First we work towards Lemma 4, which is type preservation for the reductions defined in Figure 10. Lemmas 1 and 2 enable the step from Lemma 3 to Lemma 4 in a way that corresponds to rule R-Context in Figure 10.

Lemma 1 (Typability of Subterms in E). *If $\mathcal{D}$ is a typing derivation concluding $\Gamma; \Sigma; \Phi \vdash E[e] : T$ then*

$$(\sigma;\phi;u{+}v) \longrightarrow_{\mathsf{v}} (\sigma;\phi;w) \quad \text{if } u \text{ and } v \text{ are integer literals and } u+v=w \tag{R-Plus}$$

$$\begin{array}{c}(q_0,\ldots,q_{n-1}=\alpha_0|\psi_0\rangle+\cdots+\alpha_{2^n-1}|\psi_{2^n-1}\rangle;\phi;\mathsf{measure}\ q_0,\ldots,q_{r-1}) \longrightarrow_{\mathsf{v}} \\ \boxplus_{0\leqslant m<2^r} p_m \bullet (q_0,\ldots,q_{n-1}=\frac{\alpha_{l_m}}{p_m}|\psi_{l_m}\rangle+\cdots+\frac{\alpha_{u_m}}{p_m}|\psi_{u_m}\rangle;\phi;m) \\ \text{where } l_m=2^{n-r}m,\ u_m=2^{n-r}(m+1)-1,\ p_m=|\alpha_{l_m}|^2+\cdots+|\alpha_{u_m}|^2\end{array} \tag{R-Measure}$$

$$\begin{array}{c}(q_0,\ldots,q_{n-1}=|\psi\rangle;\phi;q_0,\ldots,q_{r-1}*{=}\,U) \longrightarrow_{\mathsf{v}} (q_0,\ldots,q_{n-1}=(U\otimes I_{n-r})|\psi\rangle;\phi;\mathsf{unit}) \\ \text{where } U \text{ is a unitary operator of arity } r\end{array} \tag{R-Trans}$$

$$\begin{array}{c}(q_0,\ldots,q_{n-1}=|\psi\rangle;\phi;e) \longrightarrow_{\mathsf{v}} (q_{\pi(0)},\ldots,q_{\pi(n-1)}=\Pi|\psi\rangle;\phi;e) \\ \text{where } \pi \text{ is a permutation and } \Pi \text{ is the corresponding unitary operator}\end{array} \tag{R-Perm}$$

$$\frac{(\sigma;\phi;e) \longrightarrow_{\mathsf{v}} \boxplus_i\ p_i \bullet (\sigma_i;\phi_i;e_i)}{(\sigma;\phi;E[e]) \longrightarrow_{\mathsf{e}} \boxplus_i\ p_i \bullet (\sigma_i;\phi_i;E[e_i])} \tag{R-Context}$$

Figure 10: Reduction rules for expression configurations

$$\frac{(\sigma;\phi;e) \longrightarrow_{\mathsf{e}} \boxplus_i\ p_i \bullet (\sigma_i;\phi_i;e_i)}{(\sigma;\phi;F[e]) \longrightarrow \boxplus_i\ p_i \bullet (\sigma_i;\phi_i;F[e_i])} \tag{R-Expr}$$

$$(\sigma;\phi;c![\widetilde{v}]\,.\,P \mid c?[\widetilde{x}{:}\widetilde{T}]\,.\,Q) \longrightarrow (\sigma;\phi;P \mid Q\{\widetilde{v}/\widetilde{x}\}) \quad \text{if } |\widetilde{v}|=|\widetilde{x}| \tag{R-Com}$$

$$(\sigma;\phi;\{v\}\,.\,P) \longrightarrow (\sigma;\phi;P) \tag{R-Act}$$

$$(\sigma;\phi;(\mathsf{new}\ x{:}T)P) \longrightarrow (\sigma;\phi,c;P\{c/x\}) \quad \text{where } c \text{ is fresh} \tag{R-New}$$

$$(q_0,\ldots,q_n=|\psi\rangle;\phi;(\mathsf{qbit}\ x)P) \longrightarrow (q_0,\ldots,q_n,q=|\psi\rangle\otimes|0\rangle;\phi;P\{q/x\}) \quad \text{where } q \text{ is fresh} \tag{R-Qbit}$$

$$\frac{(\sigma;\phi;P) \longrightarrow \boxplus_i\ p_i \bullet (\sigma_i;\phi_i;P_i)}{(\sigma;\phi;P \mid Q) \longrightarrow \boxplus_i\ p_i \bullet (\sigma_i;\phi_i;P_i \mid Q)} \tag{R-Par}$$

$$\frac{P' \equiv P \quad (\sigma;\phi;P) \longrightarrow \boxplus_i\ p_i \bullet (\sigma_i;\phi_i;P_i) \quad \forall i.(P_i \equiv P_i')}{(\sigma;\phi;P') \longrightarrow \boxplus_i\ p_i \bullet (\sigma_i;\phi_i;P_i')} \tag{R-Cong}$$

$$\boxplus_i\ p_i \bullet (\sigma_i;\phi_i;P_i) \xrightarrow{p_i} (\sigma_i;\phi_i;P_i) \tag{R-Prob}$$

Figure 11: Reduction rules for process configurations

there exists U such that $\mathcal{D}$ has a subderivation $\mathcal{D}'$ concluding $\Gamma;\Sigma;\Phi \vdash e : U$ and the position of $\mathcal{D}'$ in $\mathcal{D}$ corresponds to the position of the hole in $E[\,]$.

Proof. By induction on the structure of $E[\,]$. □

Lemma 2 (Replacement in E). *If*

1. *$\mathcal{D}$ is a derivation concluding $\Gamma;\Sigma;\Phi \vdash E[e] : T$*
2. *$\mathcal{D}'$ is a subderivation of $\mathcal{D}$ concluding $\Gamma;\Sigma;\Phi \vdash e : U$*
3. *the position of $\mathcal{D}'$ in $\mathcal{D}$ matches the hole in $E[\,]$*
4. *$\Gamma;\Sigma;\Phi \vdash e' : U$*

then $\Gamma;\Sigma;\Phi \vdash E[e'] : T$.

Proof. Replace $\mathcal{D}'$ by a derivation of $\Gamma;\Sigma;\Phi \vdash e' : U$. □

Lemma 3 (Type Preservation for $\longrightarrow_{\mathsf{v}}$).
If $\Gamma;\Sigma;\Phi \vdash e : T$ and $(\sigma;\phi;e) \longrightarrow_{\mathsf{v}} \boxplus_i p_i \bullet (\sigma_i;\phi_i;e_i)$ and $\Sigma = dom(\sigma)$ and $\phi = dom(\Phi)$ then $\forall i.(\sigma_i = \sigma)$ and $\forall i.(\phi_i = \phi)$ and $\forall i.(\Gamma;\Sigma;\Phi \vdash e_i : T)$.

Proof. Examine each case in the definition of $\longrightarrow_{\mathsf{v}}$. □

Lemma 4 (Type Preservation for $\longrightarrow_{\mathsf{e}}$).
If $\Gamma;\Sigma;\Phi \vdash e : T$ and $(\sigma;\phi;e) \longrightarrow_{\mathsf{e}} \boxplus_i p_i \bullet (\sigma_i;\phi_i;e_i)$ and $\Sigma = dom(\sigma)$ and $\phi = dom(\Phi)$ then $\forall i.(\sigma_i = \sigma)$ and $\forall i.(\phi_i = \phi)$ and $\forall i.(\Gamma;\Sigma;\Phi \vdash e_i : T)$.

Proof. $(\sigma;\phi;e) \longrightarrow_{\mathsf{e}} \boxplus_i p_i \bullet (\sigma_i;\phi_i;e_i)$ is derived by the rule R-Context, so for some $E[\,]$ we have $e = E[f]$ and $\forall i.(e_i = E[f_i])$ and $(\sigma;\phi;f) \longrightarrow_{\mathsf{v}} \boxplus_i p_i \bullet (\sigma_i;\phi_i;f_i)$. From $\Gamma;\Sigma;\Phi \vdash E[f] : T$, Lemma 1 gives $\Gamma;\Sigma;\Phi \vdash f : U$ for some U, Lemma 3 gives $\forall i.(\Gamma;\Sigma;\Phi \vdash f_i : U)$ and $\forall i.(\sigma_i = \sigma)$ and $\forall i.(\phi_i = \phi)$, and Lemma 2 gives $\forall i.(\Gamma;\Sigma;\Phi \vdash E[f_i] : T)$. □

In a similar way we now work towards Theorem 1. We need substitution lemmas (10 and 11) to deal with the reduction rules R-Com, R-New and R-Qbit (Figure 11), and Lemma 12 to deal with R-Cong.

Lemma 5 (Typability of Subterms in F).
If $\mathcal{D}$ is a typing derivation concluding $\Gamma;\Sigma;\Phi \vdash F[e]$ then there exists T such that $\mathcal{D}$ has a subderivation $\mathcal{D}'$ concluding $\Gamma;\Sigma;\Phi \vdash e : T$ and the position of $\mathcal{D}'$ in $\mathcal{D}$ corresponds to the position of the hole in $F[\,]$.

Proof. By case-analysis on the structure of $F[\,]$. □

$$\Gamma \vdash v : \mathsf{Int} \quad \text{if } v \text{ is an integer literal} \qquad \Gamma \vdash \mathsf{unit} : \mathsf{Unit} \qquad \text{(T-IntLit/T-Unit)}$$

$$\Gamma \vdash \mathsf{H} : \mathsf{Op}(2) \quad \text{etc.} \qquad \Gamma, x{:}T \vdash x : T \qquad \text{(T-Op/T-Var)}$$

$$\frac{\Gamma \vdash e : \mathsf{Int} \quad \Gamma \vdash e' : \mathsf{Int}}{\Gamma \vdash e{+}e' : \mathsf{Int}} \qquad \frac{\forall i.(\Gamma \vdash x_i : \mathsf{Qbit}) \quad x_1 \ldots x_n \text{ distinct}}{\Gamma \vdash \mathsf{measure}\ x_1, \ldots, x_n : \mathsf{Int}} \qquad \text{(T-Plus/T-Msure)}$$

$$\frac{\forall i.(\Gamma \vdash x_i : \mathsf{Qbit}) \quad x_1 \ldots x_n \text{ distinct} \quad \Gamma \vdash U : \mathsf{Op}(n)}{\Gamma \vdash x_1, \ldots, x_n \mathrel{*}{=} U : \mathsf{Unit}} \qquad \text{(T-Trans)}$$

$$\Gamma \vdash \mathbf{0} \qquad \frac{\Gamma_1 \vdash P \quad \Gamma_2 \vdash Q \quad \Gamma_1 + \Gamma_2 \text{ is defined}}{\Gamma_1 + \Gamma_2 \vdash P \mid Q} \qquad \text{(T-Nil/T-Par)}$$

$$\frac{\Gamma \vdash x : \hat{}[T_1, \ldots, T_n] \quad \Gamma, y_1{:}T_1, \ldots, y_n{:}T_n \vdash P}{\Gamma \vdash x?[y_1{:}T_1, \ldots, y_n{:}T_n]\,.\,P} \qquad \frac{\Gamma, x{:}\mathsf{Qbit} \vdash P}{\Gamma \vdash (\mathsf{qbit}\ x)P} \qquad \text{(T-In/T-Qbit)}$$

$$\frac{\Gamma \vdash x : \hat{}[T_1, \ldots, T_m, \mathsf{Qbit}, \ldots, \mathsf{Qbit}] \quad \forall i.(T_i \neq \mathsf{Qbit}) \quad \forall i.(\Gamma \vdash e_i : T_i) \quad y_i \text{ distinct} \quad \Gamma \vdash P}{\Gamma, y_1{:}\mathsf{Qbit} \ldots, y_n{:}\mathsf{Qbit} \vdash x![e_1, \ldots, e_m, y_1, \ldots, y_n]\,.\,P} \qquad \text{(T-Out)}$$

$$\frac{\Gamma \vdash e : T \quad \Gamma \vdash P}{\Gamma \vdash \{e\}\,.\,P} \qquad \frac{\Gamma, x{:}\hat{}[T_1, \ldots, T_n] \vdash P}{\Gamma \vdash (\mathsf{new}\ x{:}\hat{}[T_1, \ldots, T_n])P} \qquad \text{(T-Act/T-New)}$$

Figure 12: Typing rules

$$\Gamma; \Sigma; \Phi \vdash v : \mathsf{Int} \quad \text{if } v \text{ is an integer literal} \qquad \Gamma; \Sigma; \Phi \vdash \mathsf{unit} : \mathsf{Unit} \qquad \text{(IT-IntLit/IT-Unit)}$$

$$\Gamma; \Sigma; \Phi \vdash \mathsf{H} : \mathsf{Op}(2) \quad \text{etc.} \qquad \Gamma, x{:}T; \Sigma; \Phi \vdash x : T \qquad \text{(IT-Op/IT-Var)}$$

$$\Gamma; \Sigma, q; \Phi \vdash q : \mathsf{Qbit} \qquad \Gamma; \Sigma; \Phi, c{:}T \vdash c : T \qquad \text{(IT-IdQ/IT-IdC)}$$

$$\frac{\Gamma; \Sigma; \Phi \vdash e : \mathsf{Int} \quad \Gamma; \Sigma; \Phi \vdash e' : \mathsf{Int}}{\Gamma; \Sigma; \Phi \vdash e{+}e' : \mathsf{Int}} \qquad \text{(IT-Plus)}$$

$$\frac{\forall i.(\Gamma; \Sigma; \Phi \vdash e_i : \mathsf{Qbit}) \quad \text{each } e_i \text{ is either } x_i \text{ or } q_i\text{, all distinct}}{\Gamma; \Sigma; \Phi \vdash \mathsf{measure}\ e_1, \ldots, e_n : \mathsf{Int}} \qquad \text{(IT-Msure)}$$

$$\frac{\forall i.(\Gamma; \Sigma; \Phi \vdash e_i : \mathsf{Qbit}) \quad \Gamma; \Sigma; \Phi \vdash U : \mathsf{Op}(n) \quad \text{each } e_i \text{ is either } x_i \text{ or } q_i\text{, all distinct}}{\Gamma; \Sigma; \Phi \vdash e_1, \ldots, e_n \mathrel{*}{=} U : \mathsf{Unit}} \qquad \text{(IT-Trans)}$$

$$\Gamma; \Sigma; \Phi \vdash \mathbf{0} \qquad \text{(IT-Nil)}$$

$$\frac{\Gamma_1; \Sigma_1; \Phi \vdash P \quad \Gamma_2; \Sigma_2; \Phi \vdash Q \quad \Gamma_1 + \Gamma_2 \text{ is defined} \quad \Sigma_1 \cap \Sigma_2 = \emptyset}{\Gamma_1 + \Gamma_2; \Sigma_1 \cup \Sigma_2; \Phi \vdash P \mid Q} \qquad \text{(IT-Par)}$$

$$\frac{\Gamma; \Sigma; \Phi \vdash e : \hat{}[T_1, \ldots, T_n] \quad \Gamma, y_1{:}T_1, \ldots, y_n{:}T_n; \Sigma; \Phi \vdash P}{\Gamma; \Sigma; \Phi \vdash e?[y_1{:}T_1, \ldots, y_n{:}T_n]\,.\,P} \qquad \frac{\Gamma, x{:}\mathsf{Qbit}; \Sigma; \Phi \vdash P}{\Gamma; \Sigma; \Phi \vdash (\mathsf{qbit}\ x)P} \qquad \text{(IT-In/IT-Qbit)}$$

$$\frac{\begin{array}{l}\Gamma; \Sigma; \Phi \vdash e : \hat{}[\widetilde{T}, \widetilde{\mathsf{Qbit}}] \quad \forall i.(T_i \neq \mathsf{Qbit}) \quad \forall i.(\Gamma; \Sigma; \Phi \vdash e_i : T_i) \\ \forall i.(\Gamma; \Sigma; \Phi \vdash f_i : \mathsf{Qbit}) \quad \Gamma; \Sigma; \Phi \vdash P \\ \widetilde{f} \text{ consists of distinct variables } \widetilde{f_x} \text{ and distinct qubit names } \widetilde{f_q}\end{array}}{\Gamma, \widetilde{f_x}{:}\widetilde{\mathsf{Qbit}}; \Sigma, \widetilde{f_q}{:}\widetilde{\mathsf{Qbit}}; \Phi \vdash e![e_1, \ldots, e_m, f_1, \ldots, f_n]\,.\,P} \qquad \text{(IT-Out)}$$

$$\frac{\Gamma; \Sigma; \Phi \vdash e : T \quad \Gamma; \Sigma; \Phi \vdash P}{\Gamma; \Sigma; \Phi \vdash \{e\}\,.\,P} \qquad \frac{\Gamma, x{:}\hat{}[T_1, \ldots, T_n]; \Sigma; \Phi \vdash P}{\Gamma; \Sigma; \Phi \vdash (\mathsf{new}\ x{:}\hat{}[T_1, \ldots, T_n])P} \qquad \text{(IT-Act/IT-New)}$$

Figure 13: Internal typing rules

LEMMA 6 (REPLACEMENT IN F). *If*

1. *$\mathcal{D}$ is a derivation concluding $\Gamma;\Sigma;\Phi \vdash F[e]$*
2. *$\mathcal{D}'$ is a subderivation of $\mathcal{D}$ concluding $\Gamma;\Sigma;\Phi \vdash e : T$*
3. *the position of $\mathcal{D}'$ in $\mathcal{D}$ matches the hole in $F[\,]$*
4. *$\Gamma;\Sigma;\Phi \vdash e' : T$*

then $\Gamma;\Sigma;\Phi \vdash F[e']$.

PROOF. Replace $\mathcal{D}'$ by a derivation of $\Gamma;\Sigma;\Phi \vdash e' : T$. □

LEMMA 7 (WEAKENING FOR EXPRESSIONS). *If $\Gamma;\Sigma;\Phi \vdash e : T$ and $\Gamma \subseteq \Gamma'$ and $\Sigma \subseteq \Sigma'$ and $\Phi \subseteq \Phi'$ then $\Gamma';\Sigma';\Phi' \vdash e : T$.*

PROOF. Induction on the derivation of $\Gamma;\Sigma;\Phi \vdash e : T$. □

LEMMA 8. *If $\Gamma;\Sigma;\Phi \vdash e : T$ then $fv(e) \subseteq dom(\Gamma)$ and $fq(e) \subseteq \Sigma$ and $fc(e) \subseteq dom(\Phi)$.*

PROOF. Induction on the derivation of $\Gamma;\Sigma;\Phi \vdash e : T$. □

LEMMA 9. *If $\Gamma;\Sigma;\Phi \vdash P$ then $fv(P) \subseteq dom(\Gamma)$ and $fq(P) \subseteq \Sigma$ and $fc(P) \subseteq dom(\Phi)$.*

PROOF. Induction on the derivation of $\Gamma;\Sigma;\Phi \vdash P$. □

LEMMA 10 (SUBSTITUTION IN EXPRESSIONS). *Assume that $\Gamma, \widetilde{x}{:}\widetilde{T};\Sigma;\Phi \vdash e : T$ and let $\widetilde{v}$ be values such that, for each i:*

1. *if $T_i = \mathsf{Qbit}$ then v_i is a variable or a qubit name*
2. *if $T_i = \mathsf{Qbit}$ and $v_i = y_i$ (a var) then $y_i \notin dom(\Gamma, \widetilde{x}{:}\widetilde{T})$*
3. *if $T_i = \mathsf{Qbit}$ and $v_i = q_i$ (a qubit name) then $q_i \notin \Sigma$*
4. *if $T_i \neq \mathsf{Qbit}$ then $\Gamma;\Sigma;\Phi \vdash v_i : T_i$.*

Let $\widetilde{y}$ be the variables of type Qbit from $\widetilde{v}$ (corresponding to condition (2)) and assume that they are distinct; let $\widetilde{q}$ be the qubit names from $\widetilde{v}$ (corresponding to condition (3)) and assume that they are distinct. Then $\Gamma, \widetilde{y}{:}\widetilde{\mathsf{Qbit}};\Sigma,\widetilde{q};\Phi \vdash e\{\widetilde{v}/\widetilde{x}\} : T$.

PROOF. Induction on the derivation of $\Gamma, \widetilde{x}{:}\widetilde{T};\Sigma;\Phi \vdash e : T$. □

The next lemma makes use of the addition operation on environments (Definition 1) in an essential way.

LEMMA 11 (SUBSTITUTION IN PROCESSES). *Assume that $\Gamma, \widetilde{x}{:}\widetilde{T};\Sigma;\Phi \vdash P$ and let $\widetilde{v}$ be values such that, for each i:*

1. *if $T_i = \mathsf{Qbit}$ then v_i is a variable or a qubit name*
2. *if $T_i = \mathsf{Qbit}$ and $v_i = y_i$ (a var) then $y_i \notin dom(\Gamma, \widetilde{x}{:}\widetilde{T})$*
3. *if $T_i = \mathsf{Qbit}$ and $v_i = q_i$ (a qubit name) then $q_i \notin \Sigma$*
4. *if $T_i \neq \mathsf{Qbit}$ then $\Gamma;\Sigma;\Phi \vdash v_i : T_i$.*

Let $\widetilde{y}$ be the variables of type Qbit from $\widetilde{v}$ (corresponding to condition (2)) and assume that they are distinct; let $\widetilde{q}$ be the qubit names from $\widetilde{v}$ (corresponding to condition (3)) and assume that they are distinct. Then $\Gamma, \widetilde{y}{:}\widetilde{\mathsf{Qbit}};\Sigma,\widetilde{q};\Phi \vdash P\{\widetilde{v}/\widetilde{x}\}$.

PROOF. By induction on the derivation of $\Gamma, \widetilde{x}{:}\widetilde{T};\Sigma;\Phi \vdash P$. The key cases are IT-PAR and IT-OUT.

For IT-PAR the final step in the typing derivation has the form

$$\frac{\Gamma_1;\Sigma_1;\Phi \vdash P \quad \Gamma_2;\Sigma_2;\Phi \vdash Q \quad \Gamma_1 + \Gamma_2 \text{ def.} \quad \Sigma_1 \cap \Sigma_2 = \emptyset}{\Gamma, \widetilde{x}{:}\widetilde{T};\Sigma;\Phi \vdash P \mid Q}$$

where $\Gamma_1 + \Gamma_2 = \Gamma, \widetilde{x} : \widetilde{T}$ and $\Sigma_1 \cup \Sigma_2 = \Sigma$. Each variable of type Qbit in $\Gamma, \widetilde{x}{:}\widetilde{T}$ is in exactly one of Γ_1 and Γ_2. Because the free variables of P and Q are contained in Γ_1 and Γ_2 respectively, substitution into $P \mid Q$ splits into disjoint substitutions into P and Q. The induction hypothesis gives typings for $P\{\widetilde{v}/\widetilde{x}\}$ and $Q\{\widetilde{v}/\widetilde{x}\}$, which combine (by IT-PAR) to give $\Gamma, \widetilde{y}{:}\widetilde{\mathsf{Qbit}};\Sigma,\widetilde{q};\Phi \vdash P \mid Q\{\widetilde{v}/\widetilde{x}\}$. □

LEMMA 12 (STRUCT. CONG. PRESERVES TYPING). *If $\Gamma;\Sigma;\Phi \vdash P$ and $P \equiv Q$ then $\Gamma;\Sigma;\Phi \vdash Q$.*

PROOF. Induction on the derivation of $P \equiv Q$. □

THEOREM 1 (TYPE PRESERVATION FOR $\longrightarrow$). *If $\Gamma;\Sigma;\Phi \vdash P$ and $(\sigma;\phi;P) \longrightarrow \boxplus_i p_i \bullet (\sigma_i;\phi_i;P_i)$ and $\Sigma = dom(\sigma)$ and $\phi = dom(\Phi)$ then $\forall i.(\sigma_i = \sigma)$ and $\forall i.(\phi_i = \phi)$ and $\forall i.(\Gamma;\Sigma;\Phi \vdash P_i)$.*

PROOF. By induction on the derivation of $(\sigma;\phi;P) \longrightarrow \boxplus_i p_i \bullet (\sigma_i;\phi_i;P_i)$, in each case examining the final steps in the derivation of $\Gamma;\Sigma;\Phi \vdash P$. □

THEOREM 2 (UNIQUE OWNERSHIP OF QUBITS). *If $\Gamma;\Sigma;\Phi \vdash P \mid Q$ then $fq(P) \cap fq(Q) = \emptyset$.*

PROOF. The final step in the derivation of $\Gamma;\Sigma;\Phi \vdash P \mid Q$ has the form

$$\frac{\Gamma_1;\Sigma_1;\Phi \vdash P \quad \Gamma_2;\Sigma_2;\Phi \vdash Q \quad \Gamma_1 + \Gamma_2 \text{ def.} \quad \Sigma_1 \cap \Sigma_2 = \emptyset}{\Gamma;\Sigma;\Phi \vdash P \mid Q}$$

where $\Gamma = \Gamma_1 + \Gamma_2$ and $\Sigma = \Sigma_1 \cup \Sigma_2$. By Lemma 9, $fq(P) \subseteq \Sigma_1$ and $fq(Q) \subseteq \Sigma_2$. Because $\Sigma_1 \cap \Sigma_2 = \emptyset$ we have $fq(P) \cap fq(Q) = \emptyset$. □

All of the results up to now have been proved for the internal type system (Figure 13). Our intention is that at the top level, a system should be typechecked in the original (external) type system (Figure 12), so we need the following straightforward lemma to make the connection between the two systems.

LEMMA 13 (EXTERNAL/INTERNAL TYPE SYSTEM). *$\Gamma \vdash e : T \Rightarrow \Gamma;\emptyset;\emptyset \vdash e : T$ and $\Gamma \vdash P \Rightarrow \Gamma;\emptyset;\emptyset \vdash P$.*

PROOF. Induction on the derivations. □

7. FUTURE WORK

Our aim is to develop techniques for formal verification of systems modelled in CQP. In particular we are working towards an analysis of the BB84 quantum key distribution protocol, including both the core quantum steps and the classical authentication phase. Initially we will use model-checking, in both standard (non-deterministic) and probabilistic forms. Standard model-checking is appropriate for absolute properties (for example, the quantum teleportation protocol (Section 3.2) claims that the final state of y is always the same as the initial state of z). In general, however, probabilistic model-checking is needed. For example, the

bit-commitment protocol (Section 3.3) guarantees that, with some high probability which is dependent on the number of bits used by Alice, Bob's verification step is successful. We have obtained preliminary results [16, 19] with the CWB-NC [1] and PRISM [12] systems, working directly with the modelling language of each tool. The next step is to develop automated translations of CQP into these lower-level modelling languages; note that our operational semantics matches the semantic model used by PRISM.

Another area for future work is to develop a theory of equivalence for CQP processes, as a foundation for compositional techniques for reasoning about the behaviour of systems. We can also consider extending the language. It should be straightforward to add purely classical features such as functions and assignable variables. Extensions which combine quantum data with enhanced classical control structures require more care. Valiron's [28] recent formulation of a typed quantum lambda calculus seems very compatible with our approach, and should fit into CQP's expression language.

8. CONCLUSIONS

We have defined a language, CQP, for modelling systems which combine quantum and classical communication and computation. CQP has a formal operational semantics, and a static type system which guarantees that transmitting a qubit on a communication channel corresponds to a physical transfer of ownership. The syntax and semantics of CQP are based on a combination of the pi-calculus and an expression language which includes measurement and transformation of quantum state. The style of our definitions makes it easy to enrich the language. Our research programme is to use CQP as the basis for analysis and verification of quantum protocols, and we have outlined some possibilities for the use of both standard and probabilistic model-checking.

9. REFERENCES

[1] CWB-NC: `www.cs.sunysb.edu/~cwb`.

[2] S. Abramsky and B. Coecke. A categorical semantics of quantum protocols. In *Proceedings, Nineteenth Annual IEEE Symposium on Logic in Computer Science*. IEEE Computer Society Press, 2004.

[3] C. H. Bennett and G. Brassard. Quantum Cryptography: Public-key Distribution and Coin Tossing. In *Proceedings of the IEEE International Conference on Computer, Systems and Signal Processing, Bangalore, India*, pages 175–179, 1984.

[4] C. H. Bennett, G. Brassard, C. Crépeau, R. Jozsa, A. Peres, and W. K. Wootters. Teleporting an unknown quantum state via dual classical and Einstein-Podolsky-Rosen channels. *Phys. Rev. Lett.*, 70:1895–1899, 1993.

[5] D. Cazorla, F. Cuartero, V. Valero, F. L. Pelayo, and J. J. Pardo. Algebraic theory of probabilistic and nondeterministic processes. *Journal of Logic and Algebraic Programming*, 55:57–103, 2003.

[6] H. de Riedmatten, I. Marcikic, W. Tittel, H. Zbinden, D. Collins, and N. Gisin. Long distance quantum teleportation in a quantum relay configuration. *Phys. Rev. Lett.*, 92(4), 2004.

[7] R. Ennals, R. Sharp, and A. Mycroft. Linear types for packet processing. In D. Schmidt, editor, *ESOP 2004: Proceedings of the European Symposium on Programming*, volume 2986 of *Lecture Notes in Computer Science*. Springer-Verlag, 2004.

[8] J. Gruska. *Quantum Computing*. McGraw-Hill, 1999.

[9] P. Jorrand and M. Lalire. A process-algebraic approach to concurrent and distributed quantum computation: operational semantics. In P. Selinger, editor, *Proceedings of the 2nd International Workshop on Quantum Programming Languages*, 2004. Also in Quantum Physics Archive: `arXiv:quant-ph/0407005`.

[10] E. Knill. Conventions for quantum pseudocode. Technical Report LAUR-96-2724, Los Alamos National Laboratory, 1996.

[11] N. Kobayashi, B. C. Pierce, and D. N. Turner. Linearity and the Pi-Calculus. *ACM Transactions on Programming Languages and Systems*, 21(5):914–947, September 1999.

[12] M. Z. Kwiatkowska, G. Norman, and D. Parker. PRISM: Probabilistic symbolic model checker. In T. Field, P. Harrison, J. Bradley, and U. Harder, editors, *Computer Performance Evaluation (TOOLS'02)*, pages 200–204. Springer-Verlag, 2002.

[13] D. Mayers. Unconditional Security in Quantum Cryptography. *Journal of the ACM*, 48(3):351–406, May 2001.

[14] D. A. Meyer. Quantum strategies. *Phys. Rev. Lett.*, 82(5), 1999.

[15] R. Milner, J. Parrow, and D. Walker. A calculus of mobile processes, I and II. *Information and Computation*, 100(1):1–77, September 1992.

[16] R. Nagarajan and S. J. Gay. Formal verification of quantum protocols. Quantum Physics Archive: `arXiv:quant-ph/0203086`, March 2002.

[17] M. A. Nielsen and I. L. Chuang. *Quantum Computation and Quantum Information*. Cambridge University Press, 2000.

[18] B. Ömer. Quantum programming in QCL. Master's thesis, Technical University of Vienna, 2000.

[19] N. Papanikolaou. Techniques for design and validation of quantum protocols. Master's thesis, University of Warwick, 2004.

[20] B. C. Pierce and D. Sangiorgi. Typing and subtyping for mobile processes. *Mathematical Structures in Computer Science*, 6(5), 1996.

[21] A. Poppe, A. Fedrizzi, T. Lorünser, O. Maurhardt, R. Ursin, H. R. Böhm, M. Peev, M. Suda, C. Kurtsiefer, H. Weinfurter, T. Jennewein, and A. Zeilinger. Practical quantum key distribution with polarization entangled photons. Quantum Physics Archive: `arXiv:quant-ph/0404115`, 2004.

[22] E. G. Rieffel and W. Polak. An introduction to quantum computing for non-physicists. *ACM Computing Surveys*, 32(3):300–335, 2000.

[23] P. Ryan, S. Schneider, M. Goldsmith, G. Lowe, and B. Roscoe. *Modelling and Analysis of Security Protocols*. Addison-Wesley, 2001.

[24] J. W. Sanders and P. Zuliani. Quantum programming. In *Mathematics of Program Construction*, volume 1837 of *Springer LNCS*, 2000.

[25] D. Sangiorgi and D. Walker. *The π-calculus: a Theory of Mobile Processes*. Cambridge University Press, 2001.

[26] P. Selinger. Towards a quantum programming language. *Mathematical Structures in Computer Science*, 14(4):527–586, 2004.

[27] K. Takeuchi, K. Honda, and M. Kubo. An interaction-based language and its typing system. In C. Halatsis, D. G. Maritsas, G. Philokyprou, and S. Theodoridis, editors, *PARLE '94: Parallel Architectures and Languages Europe, 6th International PARLE Conference, Proceedings*, volume 817 of *Lecture Notes in Computer Science*. Springer-Verlag, 1994.

[28] B. Valiron. Quantum typing. In P. Selinger, editor, *Proceedings of the Second International Workshop on Quantum Programming Languages*, 2004.

[29] A. van Tonder. A lambda calculus for quantum computation. *SIAM Journal on Computing*, 33(5):1109–1135, 2004.

[30] A. K. Wright and M. Felleisen. A syntactic approach to type soundness. *Information and Computation*, 115(1):38–94, 1994.